```c
 1 #include <stdio.h>
 2
 3 // vector_sum: takes two same-length vectors (double[])
 4 // adds them together component-wise in a new array
 5 // vector_sum({ 1.2, 3.4 }, {-1.0, 3.6 }) => { 0.2, 7.0 }
 6 // Assume the vectors have the same length
 7
 8 // Q: What happens if double[] is used as a return type?
 9 // double[] vector_sum(double vec1[], double vec2[]);
10
11 // Q: What about using double* as return type?
12 // double* vector_sum(double vec1[], double vec2[])
13
14 // Pass in length as an argument. Maybe now we've got it!
15 double* vector_sum(double* v1, double* v2, int len) {
16   double res[len];
17   printf("v1@%p : %p\tv2@%p : %p\tres: %p\n",
18              &v1,  v1,   &v2,  v2,     res);
19   for(int i = 0; i < len; i += 1) { res[i] = v1[i] + v2[i]; }
20   return res;
21 }
22 int main() {
23   double vec1[] = { 1.3, 4.2 }, vec2[] = { 1.5, -1 };
24   double* res1 = vector_sum(vec1, vec2, 2);
25
26   double vec3[] = { 333, 222 }, vec4[] = { 9000, 1000 };
27   double* res2 = vector_sum(vec3, vec4, 2);
28
29   printf("res1[0]: %f\t res2[0]: %f\n", res1[0], res2[0]);
30
31   printf("vec1: %p\n", vec1);
32   printf("vec2: %p\n", vec2);
33   printf("vec3: %p\n", vec3);
34   printf("vec4: %p\n", vec4);
35   printf("res1: %p\n", res1);
36   printf("res2: %p\n", res1);
37 }
```

```
$ gcc vector_sum.c -o vector_sum
vector_sum.c:22:10: warning: address of stack memory associated with
    local variable 'res' returned [-Wreturn-stack-address]
   22 |   return res;
$ ./vector_sum
v1@0x16fdcef50 : 0x16fdceff0 v2@0x16fdcef48 : 0x16fdcefe0 res: 0x16fdcef00
v1@0x16fdcef50 : 0x16fdcefd0 v2@0x16fdcef48 : 0x16fdcefc0 res: 0x16fdcef00
res1[0]: 9333.000000   res2[0]: 9333.000000
vec1: 0x16fdceff0
vec2: 0x16fdcefe0
vec3: 0x16fdcefd0
vec4: 0x16fdcefc0
res1: 0x16fdcef00
res2: 0x16fdcef00
```

Variable/Role	***Address***	***Data***							
		0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
	0x...00								
	0x...08								
	0x...10								
	0x...18								
	0x...20								
	0x...28								
	0x...30								
	0x...38								
	0x...40								
	0x...48								
	0x...50								
	0x...58								
	0x...60								
	0x...68								
	0x...70								
	0x...78								
	0x...80								
	0x...88								
	0x...90								
	0x...98								
	0x...A0								
	0x...A8								
	0x...B0								
	0x...B8								
	0x...C0								
	0x...C8								
	0x...D0								
	0x...D8								
	0x...E0								
	0x...E8								
	0x...F0								
	0x...F8								